Clase No. 6:

# Factorización LU y la librería GSL. Graficaciíon en Gnuplot

**MAT–251**

Dr. Alonso Ramírez Manzanares
CIMAT, A.C.
**e-mail:** alram@cimat.mx
**web:** http://www.cimat.mx/~alram/met_num/

Dr. Joaquín Peña Acevedo
CIMAT A.C.
**e-mail:** joaquin@cimat.mx

# Almacenamiento en memoria de la matrices

Supongamos que tenemos una matriz $m \times n$,

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

- Queremos reservar memoria para almacenar las entradas de la matriz, de manera que sea compatible con la forma en que la librería GSL almacena la información.
- También queremos que se pueda accesar mediante un arreglo bidimensional, para facilitar la forma en que accesan a las entradas de la matriz.

$$a_{ij} \qquad \rightleftarrows \qquad a[i][j]$$

# Factoriza LU usando GSL

- Para usar la librería GSL debemos utilizar las estructuras de datos que tiene definidas para vectores y matrices.
- Si ya tenemos creados arreglos 1D y 2D en la forma que establecimos, es relativamente fácil empezar a usar las estructuras de vectores y matrices de la librería GSL, sin tener que duplicar la información.

Empezamos revisando las estructuras de la librería para vectores y matrices.

# Vectores en GSL (I)

`gsl_vector` es una estructura con 5 componentes.

```
typedef struct {
    size_t size;
    size_t stride;
    double *data;
    gsl_block *block;
    int owner;
} gsl_vector;
```

El rango válido de índices es de 0 a $size - 1$. El apuntador `data` da la posición del primer elemento del arreglo.

Para crear y liberar memoria, se usan las funciones:

```
gsl_vector *gsl_vector_alloc(size_t n);
void gsl_vector_free(gsl_vector *v);
```

# Vectores en GSL (II)

Para accesar a las entradas de los vectores:

```
double gsl_vector_get(gsl_vector *v, size_t i);
void gsl_vector_set(gsl_vector *v, size_t i, double x);
```

Tambien se puede modificar los valores accesando directamente al arreglo `data`.

Ver el código *vectorGsl.c*: Lee un archivo que tiene un arreglo 1D y un escalar. Multuplica los elementos del vector por el escalar y escribe el resultado en un archivo binario.

# Matrices en GSL (I)

`gsl_matrix` es una estructura con 6 componentes.

```
typedef struct {
    size_t size1;
    size_t size2;
    size_t tda;
    double *data;
    gsl_block *block;
    int owner;
} gsl_matrix;
```

El número de filas es *size*1 y el de columnas es *size*2. El apuntador data da la posición del primer elemento del arreglo. Los datos están almacenadas por filas.

Para crear y liberar memoria, se usan las funciones:

```
gsl_matrix *gsl_matrix_alloc(size_t n1, size_t n2);
void gsl_matrix_free(gsl_matrix *m);
```

# Matrices en GSL (II)

Para accesar a las entradas de las matrices:

```
double gsl_matrix_get(gsl_matrix *m, size_t i, size_t j);
void gsl_matrix_set(gsl_matrix *m, size_t i, size_t j, double
x);
```

- El código *matricesGsl1.c* lee una matriz de un archivo y un escalar desde la línea de comandos. Usando las funciones de GSL, multiplica las entradas de la matriz por el escalar y la matriz resultante la almacena en un archivo.
- El código *matricesGsl2.c* hace lo mismo que el anterior, con la diferencia deque accesa directamente al arreglo data para modificar sus valores.

# Solución de un SEL en GSL

- El código *luGsl.c* muestra como usar la librería GSL para resolver un sistema de ecuaciones lineales usando factorización LU.
- La función *gsl_linalg_LU_decomp* que calcula la factorización contempla el intercambio de fila, de modo que además de la factorización devuelve la permutación.
- El código *luGsl2.c* lee la información de los archivos con las funciones vistas en la clase anterior y muestra como pasar esta información a las estructuras de GSL sin tener que duplicar la memoria.

# Gnuplot

- Gnuplot es programa que puede generar gráficas 2D y 3D a partir de datos proporcionados en archivos o funciones que tienen expresiones analíticas. Tambien puede hacer ajuste de datos.

- Las gráficas generadas se pueden almacenar en diferentes formatos.

- Para Windows, se puede obtener el instalador `gp501-win64-mingw.exe` y el manual de usuario en el sitio.
  `http://sourceforge.net/projects/gnuplot/files/gnuplot/5.0.1/`

- Para Ubuntu, se puede usar el administrador de paquetes para instalarlo, o ejecutar desde la linea de comandos:
  `sudo apt-get install gnuplot`

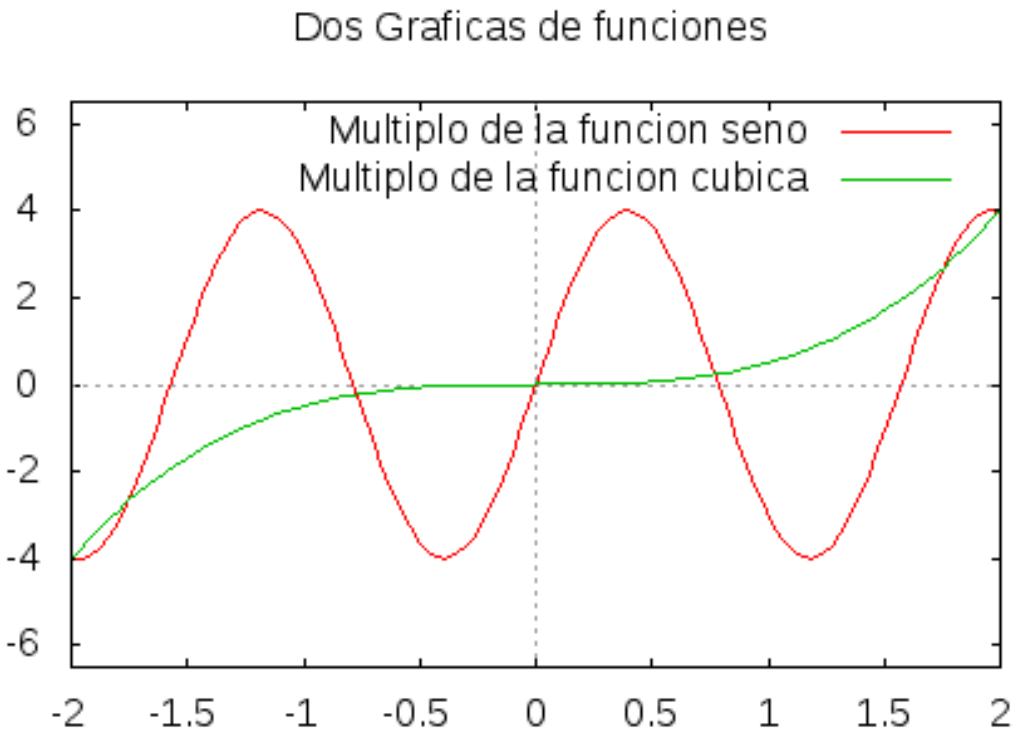
# Gnuplot

Para reproducir los ejemplos, se puede

- Mandar a ejecutar gnuplot desde una consola y dar las instrucciones dentro del ambiente de gnuplot.
- Si las instrucciones ya están en un archivo, digamos `prueba.gpl`, dentro del ambiente de gnuplot se puede ejecutar
  *load prueba.gpl*
  para generar la gráfica, o desde la línea de comandos se puede invocar el comando
  *gnuplot prueba.gpl*

# Ejemplo 2D

```
set title "Dos Graficas de funciones"
set xrange [-2:2]
set yrange [-6.5:6.5]
set zeroaxis
plot  4*sin(4*x) title 'Multiplo de la funcion seno', \
      0.5*x**3  title 'Multiplo de la funcion cubica'
```



# Para generar una imagen de salida

Para generar un archivo EPS con las graficas, hay que agregar las instrucciones:

```
# Genera un archivo EPS
set terminal postscript eps color  lw 2
set output 'graficas1.eps'
replot
set term x11
set output
```

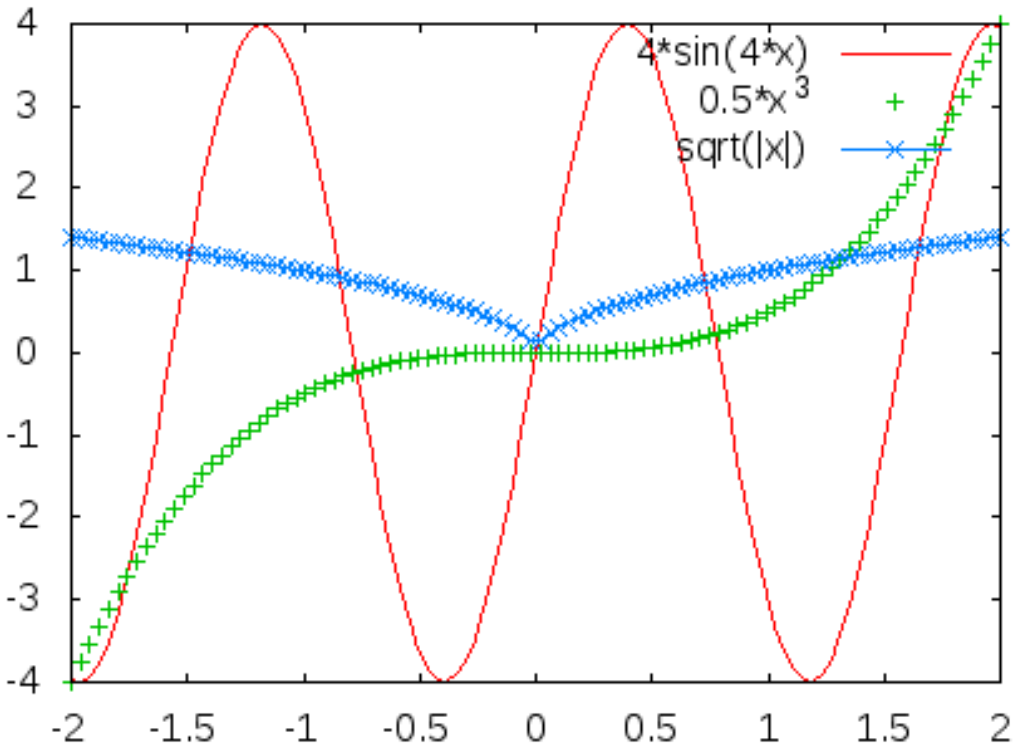
En OsX Mac viene por default "set term aqua", hay que instalar X11

Para generar un archivo PNG que tenga un tamaño de 460 pixeles de ancho y 320 pixeles de alto es:

```
# Genera un archivo PNG
set terminal png nocrop enhanced size 460,320
set output 'graficas1.png'
replot
set term x11
set output
```

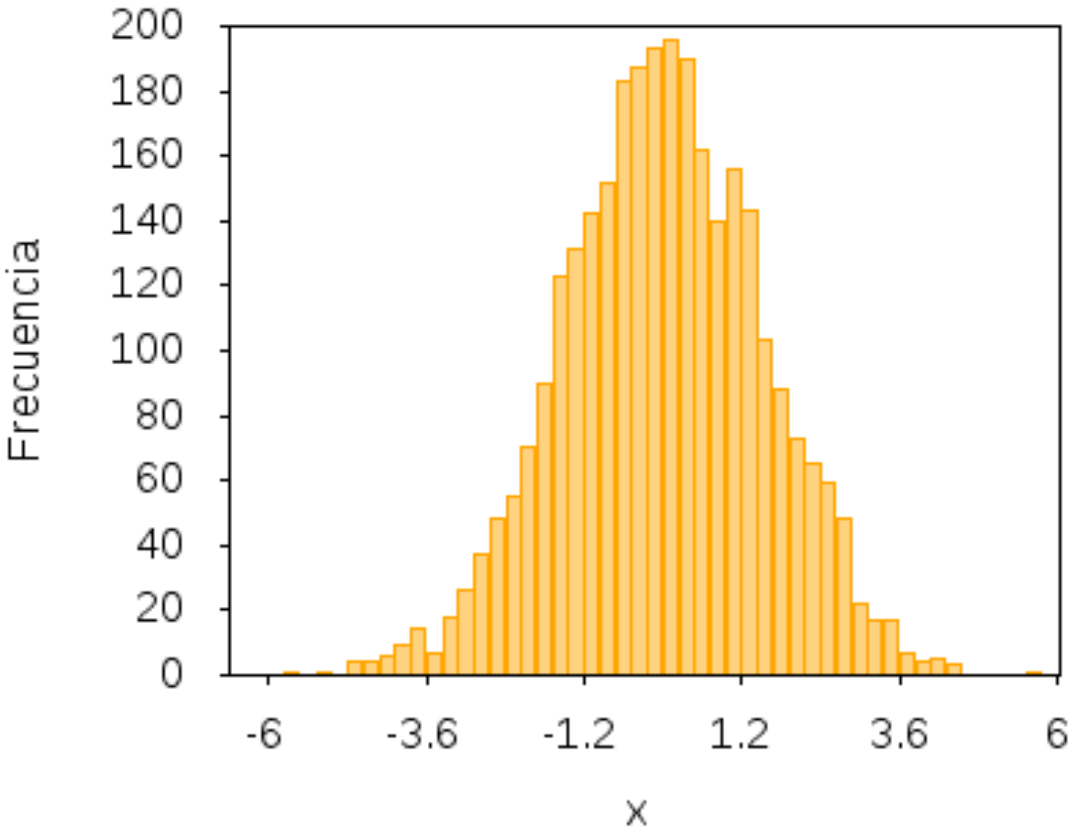
# Gráficas con datos tomados de un archivo

```
plot  "tabla.txt" using 1:2 title '4*sin(4*x)' w l, \
      "tabla.txt" using 1:3 title '0.5*x^3' w p, \
      "tabla.txt" using 1:4 title 'sqrt(|x|)' w linespoints
set terminal png nocrop enhanced size 460,320
set output 'graficas2.png'
replot
set term x11
set output
```



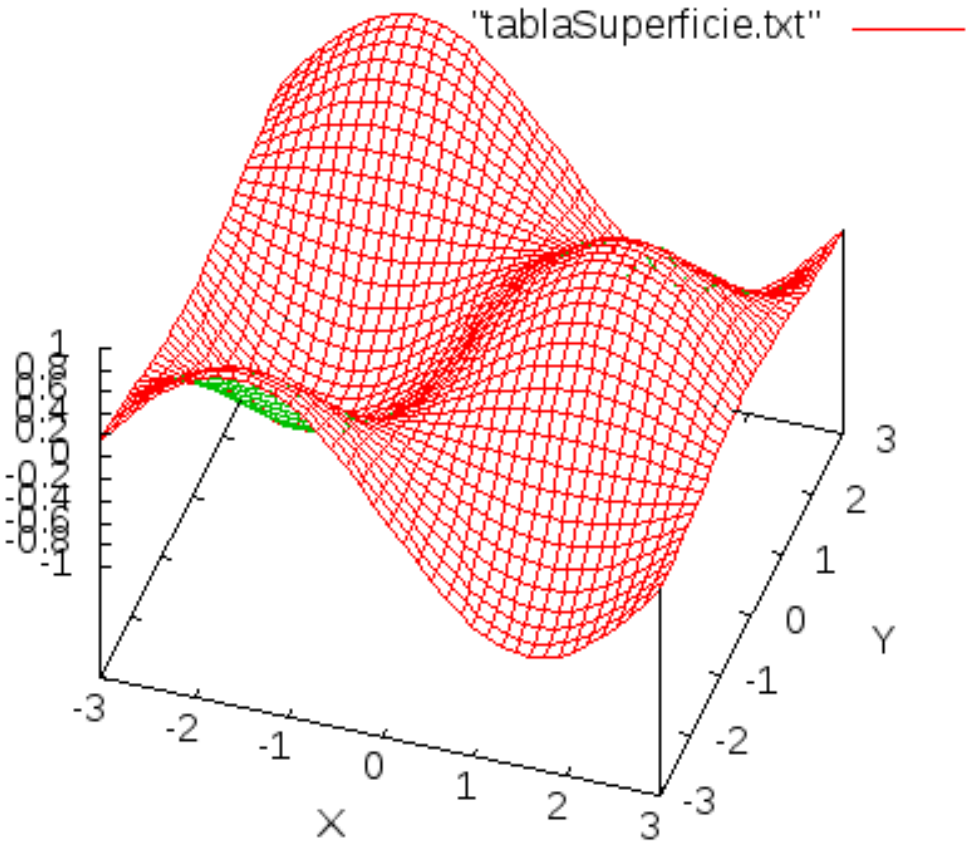
# Histograma de valores

Usando el programa *randomNormal.c*, el cual utiliza la librería GSL, genera un archivo con números muestreados de una distribución normal $N(0, \sigma)$. El programa reporta la media y la estimación de $\sigma$ y graba la muestra en un archivo de texto para visualmente verificar si su histograma corresponde a la distrubución.



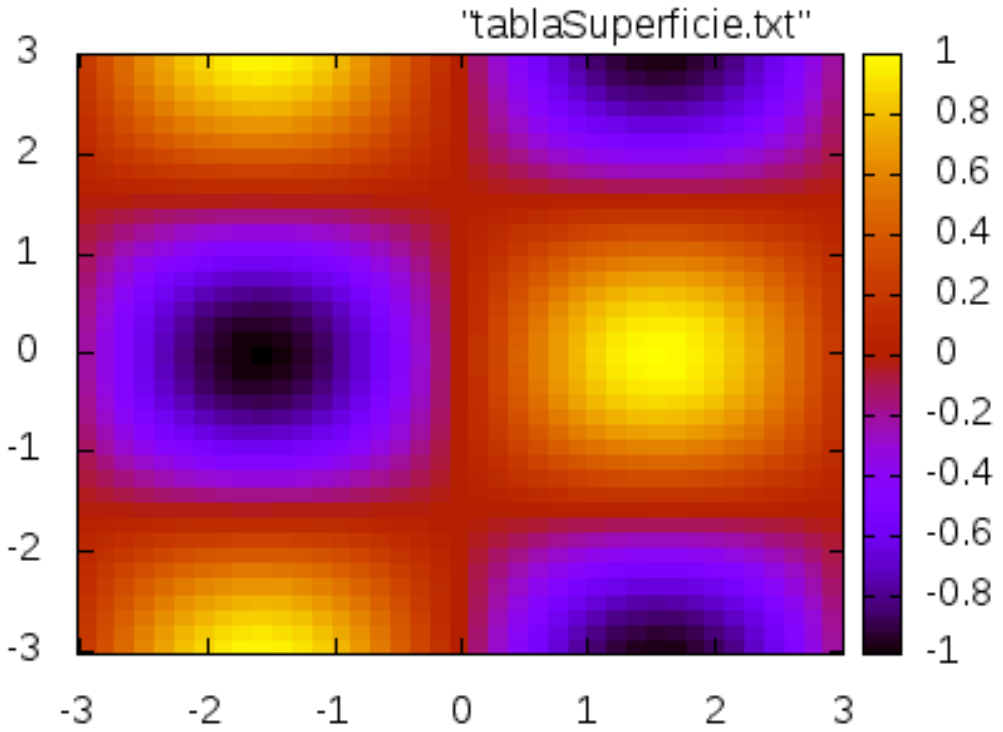
# Gráficas 3D

```
set xlabel ’X’
set ylabel ’Y’
set hidden3d
splot "tablaSuperficie.txt" with lines
```



# Gráficas 3D

```
set terminal png nocrop enhanced size 460,400
set output 'grafica4.png'
set pm3d map
splot "tablaSuperficie.txt"
set term x11
set output
```



# Gráficas 3D

Tambien se puede graficar un archivo que contiene un arreglo 2D como una imagen:

```
set terminal png nocrop enhanced size 420,320
set output 'grafica5.png'
plot './sinCoord_tablaSuperficie.txt' matrix with image
set term x11
set output
```